Subject: Re: containers access control 'roadmap'
Posted by Herbert Poetzl on Thu, 06 Sep 2007 17:10:31 GMT
View Forum Message <> Reply to Message

On Thu, Sep 06, 2007 at 11:55:34AM -0500, Serge E. Hallyn wrote:
> Roadmap is a bit of an exaggeration, but here is a list of the next bit
> of work i expect to do relating to containers and access control. The
> list gets more vague toward the end, with the intent of going far enough
> ahead to show what the final result would hopefully look like.
>
> Please review and tell me where I'm unclear, inconsistant, glossing over
> important details, or completely on drugs.
>
> 1. introduce CAP_HOST_ADMIN
>
> acts like a mask. If set, all capabilities apply across
> namespaces.
>
> is that ok, or do we insist on duplicates for all caps?
>
> brings us into 64-bit caps, so associated patches come
> along
>
> As an example, CAP_DAC_OVERRIDE by itself will mean within
> the same user namespace, while CAP_DAC_OVERRIDE|CAP_HOST_ADMIN
> will override userns equivalence checks.

what does that mean?
guest spaces need to be limited to a certain (mutable)
subset of capabilities to work properly, please explain
how this relates?

> 2. introduce per-process cap_bset
>
> Idea is you can start a container with cap-bset not containing
> CAP_HOST_ADMIN, for instance.
>
> As namespaces are fleshed out and proper behavior for
> cross-namespace access is figured out (see step 7) I
> expect behavior under !CAP_HOST_ADMIN with certain
> capabilities will change. I.e. if we get a device
> namespace, CAP_MKNOD will be different from
> CAP_HOST_ADMIN|CAP_MKNOD, and people will want to
> start keeping CAP_MKNOD in their container cap_bsets.

doesn't sound like a good idea to me, ignoring caps
or disallowing them seems okay, but changing the meaning
between caps (depending on host or guest space) seems

just wrong ...

> 3. audit driver code etc for any and all uid==0 checks. Fix those
>   immediately to take user namespaces into account.

okay, sounds good ...

> 4. introduce inode->user_ns, as per my previous userns patchset from
>   April (I guess posted in June, according to:
>   https://lists.linux-foundation.org/pipermail/containers/2007-June/005342.html)
>
> For now, enforce roughly the following access checks when
> inode->user_ns is set:
>
> if capable(CAP_HOST_ADMIN|CAP_DAC_OVERRIDE)
>   allow
> if current->userns==inode->userns {
>   if capable(CAP_DAC_OVERRIDE)
>   allow
>   if current->uid==inode->i_uid
>   allow as owner
>   inode->i_uid is in current's keychain
>   allow as owner
>   uid==inode->i_gid in current's groups
>   allow as group
> }
> treat as user 'other' (i.e. usually read-only access)

what about inodes belonging to several contexts?
(which is a major resource conserving feature of OS
level isolation)

> 5. Then comes the piece where users can get credentials as users in
>   other namespaces to store in their keychain.

does that make sense? wouldn't it be better to have
the keychains 'per context'?

> 6. enforce other userns checks like signaling
>
> 7. investigate proper behavior for other cross-namespace capabilities.

please elaborate ....